

**In the specification:**

Please substitute the following amended paragraphs for the original paragraphs having the same paragraph numbers:

{003} For example, a database product such as IBM® DB2® provides a tool with the ability to create and execute scripts which, for example, define tables to the database or obtain details of tables defined to the database.

{018} Figure 2 is a schematic diagram which shows a scripting tool which can create scripts for performing actions against a UDDI registry according to the preferred embodiment of the present invention. The scripting tool 200 provides a graphical interface which enables a user to define and execute actions which result in SOAP requests being sent to a UDDI registry. The tool includes a user ~~input/output~~ input/output and processing component 201 which accepts and processes user requests to execute actions and reports results of executed actions to the user. The user i/o and processing component further accepts user inputs which specify that a script is to be created and indication of a scripting language to be used when creating the script. The tool further includes a SOAP component 202 which is used by the user i/o and processing component to access (251) the UDDI registry 212 via the registry SOAP component 211. A user with a task to complete specifies to the user i/o and processing component an action to be executed against the UDDI registry, which the tool does by converting the action to an appropriate SOAP request and sending it to the UDDI registry. A response is then received from the UDDI registry for which the scripting tool reports the result back to the user. Depending on the result the user may

then define and execute one or more further requests until the task to be completed is finished. The user then indicates that the task is complete. During this process a macro record component 203 maintains a record of the sequence of actions the user executed and so, when the user indicates that the task is complete, the user i/o and processing component asks the user if a script containing the recorded sequence of actions should be created. The user is further given a choice of scripting language for the script to be created. In the preferred embodiment the languages available are the scripting languages PERL SCRIPT and JAVASCRIPT PerlScript and JavaScript. If the user chooses (252) the PERL SCRIPT scripting language, a PERL SCRIPT PerlScript a PerlScript engine 204 is used to generate a script 206 205 in the PERL SCRIPT scripting PerlScript language, and if the user chooses (253) the JAVASCRIPT scripting language, a JAVASCRIPT JavaScript script a JavaScript engine 205 206 is used to generate a script 207 in the JAVASCRIPT scripting JavaScript language.

{021} Once these two actions have been completed the user indicates that the task is complete and is then given the choice of creating a script which can be used to repeat the sequence of actions. For example if the user decides to generate a JAVASCRIPT scripting language JavaScript script the tool may output a script of the form:

```
...  
var findBizReq = new FindBusinessRequest();  
findBizReq.setName("IBM");  
findBizReq.addFindQualifier("caseSensitiveMatch");  
...  
var businessList = proxy.find_business(findBizReq);
```

...

```
var getBusinessDetailReq = new GetBusinessDetailRequest(),  
getBusinessDetailReq.addBusinessKey("11E5AE0F-230C-4889-8E0D-023F"),  
var businessDetail = proxy.get__businessDetail(getBusinessDetailReq);
```

{022} Alternatively, for example, if the user decides to generate a PERLSCRIPT scripting language PerlScript script the tool may output a script of the form:

```
#!/usr/bin/perl  
  
$findBizReq = CreateBean("org.uddi4j.adapt.FindBusinessRequest");  
$findBizReq->addName("IBM");  
$findBizReq->addFindQualifier("caseSensitiveMatch");  
...  
$businessList = $proxy->find__business($findBizReq);  
...  
$getBusinessDetailRequest = CreateBean("org.uddi4j.adapt.GetBusinessDetailRequest");  
$getBusinessDetailRequest->addBusinessKey("11E5AE0F-230C-4889-8E0D-023F");  
$businessDetail = proxy->get__businessDetail($getBusinessDetailRequest);
```

{023} A created script may then be run using an appropriate tool which is independent of the tool used to create the script. Returning to figure 2, the PERLSCRIPT scripting language PerlScript script 206 may subsequently be executed (254) using an execution tool which provides a PERLSCRIPT scripting language PerlScript engine 231 and a SOAP component 232.

Alternatively the JavaScript script 207 may subsequently be executed (255) ~~run~~ on an execution tool which provides a JAVASCRIPT scripting language JavaScript engine 221 and a SOAP component 222. In both cases the appropriate engine converts each action in the script into a SOAP request for sending (256) to the UDDI registry 212.

{025} Note that whilst the tool of the preferred embodiment only supports JAVASCRIPT and PERL SCRIPT scripting language scripts, JavaScript and PerlScript scripts in another embodiment it is could support more or different scripting languages such as NETREXX, BEAN MARKUP LANGUAGE (BML), JACL, ADVENTURE CREATION LANGUAGE, JYTHON, PYTHON, VBSCRIPT, JSCRIPT and PERLSCRIPT ~~NetRexx, Bean Markup Language (BML), JACL Adventure Creation Language, Jython, Python, VBScript, Jscript and PerlScript.~~

{026} Further note that a skilled person would realize ~~realise~~ that the method described in figure 3 could be implemented in a variety of programming languages, for example, the programming languages JAVA Java™, C, and C++ (JAVA Java and all JAVA-based Java-based trademarks are trademarks of SUN MICROSYSTEMS Sun Microsystems, Inc. in the United States, other countries, or both). Further a skilled person would realize ~~realise~~ that once implemented the methods can be stored in a computer program product comprising or more programs, in source or executable form, on a media, such as floppy disk, CD, and DVD, suitable for loading onto a data processing host and causing the data processing host to carry out the methods.